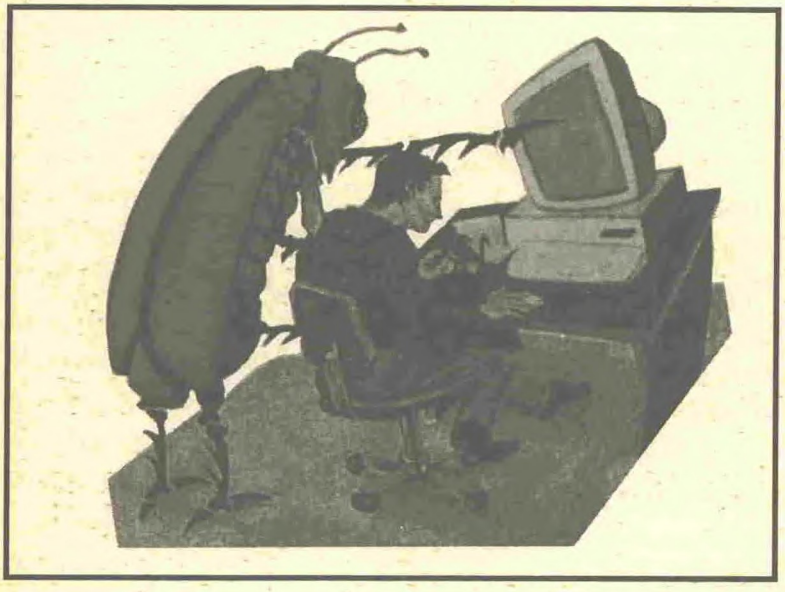
สุทธิศักดิ์ พงศ์ธนาพาณิช

Advanced BASIC Programming Technique ตอน เรียนรู้คลาส (Class) ของวินโดวส์ 95



เรียนรู้ความหมายและวิธีการใช้งานคลาสของ VB/Win

จากตอนที่ผ่านมาผู้เขียนได้แนะนำพื้นฐานเกี่ยวกับวินโดวส์ 95 และการโปรแกรมฟังก์ชันวินโดว์ API ระดับ 32 บิตด้วย VB/Win ไปบ้างแล้ว ซึ่งจะทำให้ผู้อ่านเข้าใจถึงโครงสร้างพื้นฐานของการจัดส่งพารามิเตอร์ให้กับฟังก์ชันวินโดว์ API ทั้งหมด สำหรับในบทความตอนนี้ก็จะเป็นเรื่องของคลาส (Class) ซึ่งคำว่าคลาสอาจจะไม่เป็นที่คุ้นเคยนักสำหรับโปรแกรมเมอร์ VB/Win เพราะ VB/Win เริ่มที่จะสนับสนุนการสร้างออบเจ็กต์โดยใช้คลาสในเวอร์ชัน 4.0 นี่เอง (การใช้โมดูล .CLS ในการสร้างออบเจ็กต์ต้นแบบ) และก็มีผู้อ่านหลาย ๆ ท่านยังไม่เข้าใจถึงความหมายและวิธีการใช้งานคลาสของ VB/Win สำหรับผู้อ่านที่สนใจการเขียนโปรแกรมเชิงวัตถุโดยใช้คลาสของ VB/Win 4.0 ก็ให้ติดตามได้จากบทความชื่อ Object Oriented Programming ด้วย VB/Win

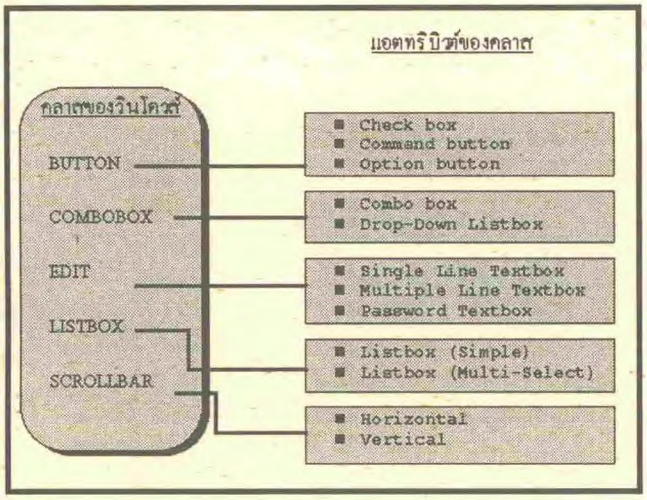
ในความเป็นจริงคลาสของวินโดวส์กับคลาสของ VB/Win นั้นไม่เหมือนกันเสียทีเดียว เพราะวินโดวส์จะจัดการกับทุก ๆ ออบเจ็กต์โดยกำหนดให้ทุก ๆ หน้าต่างเป็นสมาชิกของคลาสหนึ่ง ๆ แต่สำหรับ VB/Win จะจัดการกับคลาสของวินโดว์ต่าง ๆ ให้ด้วยตัวเอง โดยที่คลาส (.CLS) ของ VB/Win นั้นเป็นเพียงพิมพ์เขียวอย่างหนึ่งที่อนุญาตให้ผู้อ่านสามารถสร้างออบเจ็กต์ (ไม่สามารถแสดงผลเพราะไม่ใช่หน้าต่าง) ที่ประกอบด้วยคุณสมบัติ (property) และวิธี (method) เท่านั้น

สำหรับตัวผู้เขียนแล้วเห็นว่า จำเป็นที่ผู้อ่านที่เป็นโปรแกรมเมอร์ VB/Win นั้น ควรที่จะทำความเข้าใจเกี่ยวกับแนวคิดของคลาสของวินโดวส์ 95 เพราะในการจัดการหรือควบคุมหน้าต่างทั้งหลายหรือส่วนของแอพพลิเคชันอื่น ๆ ที่ทำงานภายใต้วินโดวส์ 95 นั้น จำเป็นต้องกระทำผ่านทางคลาสเท่านั้น ซึ่งความสามารถเหล่านี้ก็ไม่มีใน VB/Win อีกด้วย ดังนั้นถ้าหากผู้อ่านทราบถึงวิธีการจัดการและติดต่อกับคลาสของวินโดวส์ 95 อย่างสมบูรณ์แล้ว ก็จะทำให้ผู้อ่านสามารถควบคุมแอพพลิเคชันทั้งหมดของวินโดวส์ได้ด้วยโปรแกรมเพียงโปรแกรมเดียว หรือที่สำคัญผู้อ่านยังสามารถแกะรอยองค์ประกอบของเอนไวรอนเมนต์ VB/Win ได้อย่างไม่มีข้อจำกัดอีกด้วย ผู้อ่านที่เคยอ่านหนังสือหรือบทความต่างประเทศบางท่านคงสงสัยว่า ทำไมโปรแกรมเมอร์ต่างประเทศจึงทราบว่า VB/Win (VB.EXE) นั้นมีชื่อคลาสว่า ThunderMain และหน้าต่างโปรเจ็กต์ของ VB/Win ก็มีชื่อคลาสว่า PROJECT เขาทราบกันอย่างไร? ซึ่งผู้อ่านจะหมดความสงสัยทันทีที่อ่านบทความตอนนี้จบ และนอกจากนี้แล้วผู้อ่านยังสามารถทราบถึงคลาสของโปรแกรมอื่น ๆ ได้ด้วยตนเองอีกด้วย ไม่ยกเว้นแม้กระทั่งตัวของวินโดวส์ 95 เอง

คลาสของวินโดวส์ 95

ตามแนวคิดของระบบปฏิบัติการวินโดวส์ ทุก ๆ แอพพลิเคชันหรือองค์ประกอบของแอพพลิเคชันล้วนแต่เป็นหน้าต่าง (วินโดว์) หนึ่ง ๆ เท่านั้น ที่จะถูกควบคุมอยู่ภายใต้วินโดว์ใหญ่ ซึ่งก็คือตัววินโดวส์เอง (โดยปกติเราเรียกว่า desktop) โดยที่หน้าต่างทั้งหมดนี้ก็จะต้องเป็นสมาชิกของคลาสหนึ่ง ๆ เช่น ไมโครซอฟต์เวิร์ด เวอร์ชัน 7.0 สำหรับวินโดวส์ 95 ก็จะมีชื่อคลาสว่า _sdm_Microsoft word for Windows 95 เป็นต้น สำหรับองค์ประกอบของ VB/Win ก็มีคลาสเช่นกัน เช่น ฟอร์ม คอนโทรล เช่น Picture box, Text box, List box หรือ Label เป็นต้น และนอกจากนี้ออบเจ็กต์ที่อยู่ภายใต้คลาสเดียวกันอาจจะมีการแสดงผลหรือพฤติกรรมที่แตกต่างกันก็ได้ ทั้งนี้ก็ขึ้นกับแอตทริบิวต์ (attribute) ที่เรากำหนดให้กับแต่ละออบเจ็กต์นั่นเอง สำหรับคลาสมาตรฐานของวินโดวส์ 95 ก็มีดังต่อไปนี้

ชื่อคลาส	รายละเอียด
BUTTON	ถูกใช้โดยออบเจ็กต์ชนิด command button, option button และ check box
COMBOBOX	ถูกใช้โดยออบเจ็กต์ชนิด combo box ซึ่งเป็นการรวมระหว่าง edit box กับ list box
EDIT	ถูกใช้โดยออบเจ็กต์ชนิด edit box ทั้งชนิดบรรทัดเดียวและหลาย ๆ บรรทัด
LISTBOX	ถูกใช้โดยออบเจ็กต์ชนิด list box
SCROLLBAR	ถูกใช้โดยออบเจ็กต์ชนิด scroll bar ทั้งชนิดแนวตั้งและแนวนอน
STATIC	ถูกใช้โดยออบเจ็กต์ชนิด หน้าต่างแสดงผลข้อความ
MDICLIENT	ถูกใช้โดยออบเจ็กต์ชนิด ฟอร์ม MDI (multiple document interface)



รูปที่ 1 แสดงคลาสมาตรฐานของวินโดวส์

เมื่อผู้อ่านต้องการที่จะกำหนดให้ออบเจ็กต์ (ต่อไปจะขอเน้นเฉพาะกับหน้าต่างเท่านั้น) ใด ๆ เป็นสมาชิกของคลาสใดคลาสหนึ่ง ผู้อ่านก็ต้องมีการกำหนดแอตทริบิวต์ให้กับหน้าต่างนั้น ๆ ด้วย เพื่อเป็นการกำหนดคุณลักษณะเฉพาะของวินโดวส์ ซึ่งคุณลักษณะเฉพาะของวินโดวส์จะมีผลต่อหน้าต่างเฉพาะในขณะทำงาน (run time) เท่านั้น เพราะคลาสจะถูกโหลดขึ้นมายังหน่วยความจำและมีการเปลี่ยนแปลงแอตทริบิวต์เฉพาะในหน่วยความจำเท่านั้น จะไม่แตะต้องไฟล์ต้นฉบับของแอพพลิเคชันที่เป็นเจ้าของหน้าต่างนั้น ๆ โดยที่คลาสแอตทริบิวต์มาตรฐานของวินโดวส์ 95 ก็มีดังต่อไปนี้

แอตทริบิวต์ของคลาส	รายละเอียด
Background	กำหนดสีที่ใช้แสดงผลพื้นหลังของหน้าต่างแต่ละคลาส
Class Extra Bytes	ไบต์ข้อมูลที่จัดเก็บข้อมูลที่สำหรับคลาสหนึ่ง ๆ จะนำไปใช้งานร่วมกัน
Class Function	ฟังก์ชันที่ถูกกำหนดเป็น default ของคลาสนั้น ๆ
Class Style	กำหนดคุณลักษณะต่าง ๆ ของหน้าต่างที่เป็นสมาชิกของคลาสนั้น ๆ
Cursor	กำหนดเคอร์เซอร์ที่เป็น default เมื่อมีการเลื่อนเม้าส์ให้อยู่ภายในพื้นที่ของหน้าต่างที่เป็นสมาชิกของคลาสนั้น ๆ
Icon	กำหนดไอคอนที่เป็น default สำหรับแสดงผลเมื่อมีการ minimized หน้าต่างที่เป็นสมาชิกของคลาสนั้น ๆ
Instance	กำหนด instance ที่เป็นเจ้าของคลาส ซึ่งคลาสทั้งหลายของ VB/Win มักจะเป็นคลาสของวินโดวส์หรือรันไทม์ไลบรารีของ VB/Win (VB40032.DLL)
Menu	กำหนด menu ที่เป็น default ของวินโดวส์ที่เป็นสมาชิกของคลาสนั้น ๆ

คลาสที่วินโดวส์สนับสนุนนั้นสามารถแบ่งออกได้เป็น 2 กลุ่มหลัก ๆ ด้วยกันดังนี้ คลาสเฉพาะของแอพพลิเคชัน ซึ่งเป็นคลาสที่แต่ละแอพพลิเคชันจัดการด้วยตัวเองและจะไม่ถูกลงทะเบียน (register) เมื่อมีการโหลดหรือสิ้นสุดการทำงานของแอพพลิเคชัน ดังนั้นคลาสชนิดนี้จะส่งผลเฉพาะในขณะที่มีการทำงานของแอพพลิเคชันเท่านั้น ส่วนคลาสชนิดโกลบอล ก็เป็นคลาสของวินโดวส์ที่สามารถใช้งานร่วมกันได้โดยหลาย ๆ แอพพลิเคชัน และวินโดวส์จะทำการลงทะเบียนทุกครั้งที่มีการโหลดวินโดวส์หรือแอพพลิเคชัน ดังนั้นถ้าหากแอพพลิเคชันที่ถูกสร้างขึ้นมามีการเปลี่ยนแปลงแอตทริบิวต์ของคลาสชนิดโกลบอล ผู้อ่านก็ต้องออกจากวินโดวส์และโหลดวินโดวส์ใหม่ทุกครั้ง เพื่อกำหนดให้วินโดวส์ลงทะเบียน และทำการโหลดแอตทริบิวต์ของคลาสขึ้นมาใช้ใหม่อีกครั้ง ยกตัวอย่างเช่น เมื่อมีการติดตั้ง VB/Win โปรแกรม Setup ก็จะบังคับให้มีการออกจากวินโดวส์และกลับเข้ามาใหม่ทุกครั้ง เพื่อเป็นการ refresh คลาสที่ถูกแก้ไขนั่นเอง

สำหรับชื่อคลาสของคอนโทรล .OCX มาตรฐานของ VB/Win นั้นผู้อ่านสามารถตรวจสอบได้โดยใช้โปรแกรมตัวอย่าง winclass.vbp สำหรับตัวอย่างชื่อคลาสของคอนโทรล .OCX ที่ผู้เขียนได้ทำการตรวจสอบแล้วก็มีดังนี้

ชื่อคอนโทรล	ชื่อคลาส
Check box	ThunderCheckBox
Combo box	ThunderComboBox
Form	ThunderForm
Frame	ThunderFrame
Label	ThunderLabel
MDIForm	ThunderMDIForm
Picture box	ThunderPictureBox
Text box	ThunderTextBox
Timer	ThunderTimer

Z-Order

ในการสร้างแอพพลิเคชันหนึ่งโดยปกติจะประกอบด้วยฟอร์มหลาย ๆ ฟอร์ม และในการแสดงผลหนึ่ง ๆ นั้น ผู้อ่านสามารถแสดงผลพร้อม ๆ กันได้ครั้งละหลาย ๆ ฟอร์ม หรือแม้กระทั่งแอพพลิเคชันชนิด MDI ซึ่งจะประกอบด้วยฟอร์มลูก (child form) หลาย ๆ ภายในพื้นที่แสดงผลของฟอร์มแม่ (parent form) ฟอร์มเดียว (ดูตัวอย่างได้จากโปรแกรมตัวอย่าง MDINOTE.VBP ของ VB/Win) ก็เช่นเดียวกัน เมื่อหลาย ๆ ฟอร์มที่แสดงผลมีการซ้อนทับกัน วินโดวส์ก็ต้องมีหน้าที่จัดลำดับการแสดงผลของแต่ละฟอร์ม ซึ่งวิธีที่วินโดวส์ใช้จัดการนี้เราเรียกว่า Z-order ซึ่งหลักการณ์ของ Z-order มีดังนี้ สมมติว่าระนาบหน้าจอแสดงผลเป็นแกน x-y และกำหนดให้แกนที่พุ่งออกจากระนาบหน้าจอแสดงผลเป็นแกน z และหน้าต่างทั้งหลายที่ซ้อน ๆ กันนั้น ก็จะมีการจัดเรียงลำดับตามแนวแกน z ดังนั้นถ้าหากหน้าต่างใดอยู่บนสุดหรือปลายสุดของแกน z ก็จะสามารถแสดงผลได้สมบูรณ์ ส่วนหน้าต่างที่อยู่ล่าง ๆ ลงมาก็จะไม่สามารถแสดงผลได้หมด โดยวินโดวส์จะแสดงผลเฉพาะส่วนที่ไม่ถูกซ้อนทับโดยหน้าต่างด้านบนเท่านั้น และก็เป็นเช่นนี้เรื่อย ๆ จนถึงหน้าต่างที่อยู่ล่างสุด ซึ่งเราเรียกหลักการณ์ซ้อนเรียงหน้าต่างนี้ว่า Z-Order

การจัดการหน้าต่างของวินโดวส์

จากตอนที่ผ่าน ๆ มาเกี่ยวกับฟังก์ชัน API สำหรับวินโดวส์ 3.x นั้น จะเห็นว่าฟังก์ชันโดยส่วนใหญ่ต้องการพารามิเตอร์ที่เป็นหมายเลข handle ดังนั้นเพื่อความเข้าใจถึงความหมายของ handle ก็จะขออธิบายลักษณะและวิธีการใช้งานของหมายเลข handle ในหัวข้อนี้ก็แล้วกัน

ระบบปฏิบัติการวินโดวส์มีการมองแต่ละส่วนไม่ว่าจะเป็นไฟล์ .DLL, .VDx, .EXE หรือ .OCX ว่าเป็นออบเจ็กต์หนึ่ง ๆ ของวินโดวส์ และในการโหลดแต่ละออบเจ็กต์ วินโดวส์ก็จะกำหนดหมายเลข ID ให้กับแต่ละออบเจ็กต์โดยที่หมายเลข ID นี้จะไม่ซ้ำกันเลย ซึ่งเราเรียกหมายเลข ID นี้ว่า handle ซึ่งเป็นตัวเลขจำนวนเต็ม long (4 ไบต์) และก็เช่นเดียวกันทุก ๆ หน้าต่างที่ถูกโหลดโดยวินโดวส์ ก็จะมีหมายเลข handle เช่นเดียวกัน ดังที่จะเห็นได้จากพารามิเตอร์ HWnd ของประโยคการประกาศฟังก์ชัน และสำหรับคอนโทรลของ VB/Win นั้นโดยปกติก็จะมีคุณสมบัติ hWnd เพื่อรายงานหมายเลข handle ของคอนโทรลนั้น ๆ อยู่แล้ว แต่ก็มีบางคอนโทรลที่ไม่มีคุณสมบัติดังกล่าว ซึ่งในกรณีนี้ก็สามารถอ่านหมายเลข handle ของคอนโทรลนั้นได้จากฟังก์ชันวินโดว์ API ที่ชื่อ GetFocus

จากรายละเอียดข้างต้นได้กล่าวคร่าว ๆ ถึงเรื่องของคลาสไปบ้างแล้วพอสมควร ดังนั้นในส่วนต่อไปของบทความตอนนี้ ก็จะเขียนถึงวิธีการใช้งานฟังก์ชันที่เกี่ยวข้องกับเรื่องคลาสของออบเจ็กต์ของวินโดวส์ และโปรแกรมตัวอย่างการใช้งานฟังก์ชันเหล่านี้ เพื่อให้ผู้อ่านสามารถเข้าใจถึงวิธีการใช้งานฟังก์ชันที่เกี่ยวข้องกับคลาสของวินโดวส์ได้ดียิ่งขึ้น

GetClassName

การประกาศ

```
Public Declare Function GetClassName Lib "user32" _
Alias "GetClassNameA" (ByVal hwnd As Long, _
ByVal lpClassName As String, ByVal nMaxCount As Long) _
As Long
```

หน้าที่ รายงานชื่อคลาส ของหน้าต่างที่ถูกกำหนด

พารามิเตอร์

hwnd	หมายเลข handle ของหน้าต่างที่ต้องการ
lpClassName	สตริงที่มีขนาดเท่ากับ 256 ไบต์ เพื่อใช้เป็นบัฟเฟอร์สำหรับจัดเก็บชื่อคลาสของหน้าต่างที่ต้องการ
nMaxCount	กำหนดขนาดของบัฟเฟอร์ lpClassName - 1 (ในที่นี้เท่ากับ 255 ไบต์)

ตัวอย่างการใช้

```
' next, add the class name
    buf$ = String$(256, 0)
    Dummy& = GetClassName(hwnd&, buf$, 255)
    desc$ = desc$ & sZtrim$(buf$) & vbTab
```

ค่าที่ส่งกลับ

> 0	หมายถึง ขนาดความยาวของชื่อคลาส ของหน้าต่างที่กำหนด
= 0	หมายถึง เกิดข้อผิดพลาดในการทำงานของฟังก์ชัน

หมายเหตุ ในการกำหนดขนาดของบัฟเฟอร์สตริงนั้น ผู้อ่านต้องกำหนดให้มีขนาดโตกว่าขนาดของสตริงที่ต้องการอ่านอยู่ 1 ไบต์ ทั้งนี้เนื่องจากสตริงที่รับนั้นจะอยู่ในรูปแบบของภาษาซี ซึ่งจะมี ascii 0 (null terminate string) ต่อท้ายเสมอ

GetCurrentProcessId

การประกาศ Public Declare Function GetCurrentProcessId Lib _
"kernel32" () As Long

หน้าที่ รายงานหมายเลข ID ของขบวนการ (process) ปัจจุบัน

พารามิเตอร์ ไม่มี

ตัวอย่างการใช้
```
' get thread ID of process next, and compare.
            Dummy& = GetWindowThreadProcessId(hwnd, hWndProcess)
     If hWndProcess = GetCurrentProcessId() Then
```

ค่าที่ส่งกลับ หมายเลข ID ของขบวนการปัจจุบัน

หมายเหตุ สำหรับรายละเอียดของคำว่า ขบวนการ (process) สำหรับแอพพลิเคชันที่ทำงานบนวินโดวส์ 95 (Win32c) ให้ผู้อ่านศึกษาได้จากบทความตอนที่ผ่านมา

GetDesktopWindow

การประกาศ Public Declare Function GetDesktopWindow Lib _
"user32" () As Long

หน้าที่ รายงานหมายเลข handle ของหน้าต่างที่ทำหน้าที่แสดงผลแทนทั้งจอภาพ (desktop)

พารามิเตอร์ ไม่มี

ตัวอย่างการใช้
```
' The desktop is the highest window.
hwnd& = GetDesktopWindow()
```

ค่าที่ส่งกลับ หมายเลข handle ของหน้าต่างที่ทำหน้าที่แสดงผลแทนทั้งจอภาพ

หมายเหตุ สำหรับหน้าต่างที่ทำหน้าที่แสดงผลแทนทั้งจอภาพ (desktop) นั้น คือพื้นหลังของวินโดวส์ที่แสดงผลบนจอภาพนั่นเอง ซึ่งไอคอนทุก ๆ ไอคอนจะต้องแสดงผลในหน้าต่างนี้ และเนื่องจากวินโดวส์จัดการกับหน้าต่างทั้งหมดที่แสดงผลในลักษณะของ link-list โดยกำหนดให้หน้าต่าง desktop เป็นหน้าต่างลำดับสูงสุดของ link-list ดังนั้นก่อนที่ผู้อ่านจะอ่านข้อมูลจากหน้าต่างต่อ ๆ ไป ผู้อ่านต้องมีการอ่านหมายเลข handle โดยเริ่มต้นจากหน้าต่าง desktop นี้ทุกครั้ง

GetFocus

การประกาศ Public Declare Function GetFocus Lib "user32" () As Long

หน้าที่ รายงานหมายเลข handle ของหน้าต่างที่กำลังรับ input

พารามิเตอร์ ไม่มี

ตัวอย่างการใช้
```
hwndInput& = GetFocus()
```

ค่าที่ส่งกลับ
- <> 0 หมายถึง หมายเลข handle ของหน้าต่างที่กำลังรับ input (get focus)
- = 0 หมายถึง ไม่มีหน้าต่างที่อยู่ในสถานะได้รับ focus ในขณะนั้น

หมายเหตุ ไม่มี

GetModuleFileName

การประกาศ Public Declare Function GetModuleFileName Lib "kernel32" _
Alias "GetModuleFileNameA" (ByVal hModule As Long, _
ByVal lpFileName As String, ByVal nSize As Long) As Long

หน้าที่ รายงานชื่อไฟล์ (รวมทั้งชื่อไดเรคทอรีด้วย) ของโมดูลที่ถูกกำหนด

พารามิเตอร์
- hModule หมายเลข handle ของโมดูล .DLL หรือ .EXE ที่กำหนด
- lpFileName สตริงที่มีขนาดเท่ากับ 256 ไบต์ เพื่อใช้เป็นบัฟเฟอร์สำหรับจัดเก็บชื่อไฟล์ของโมดูลที่รายงานโดยฟังก์ชันนี้
- nSize กำหนดขนาดของบัฟเฟอร์ lpFileName - 1 (ในที่นี้เท่ากับ 255 ไบต์)

ตัวอย่างการใช้
```
' Get the module filename.
Dummy& = GetModuleFileName(inst&, buf$, 255)
buf$ = sZtrim$(GetBaseName(buf$))
```

ค่าที่ส่งกลับ
- > 0 หมายถึง ความยาวชื่อไฟล์ของโมดูลที่อ่านได้
- = 0 หมายถึง เกิดข้อผิดพลาดในขณะที่ฟังก์ชันทำงาน

→

หมายเหตุ สำหรับชื่อไฟล์ (รวมทั้งชื่อไดเรคทอรี) นั้น จะถูกแยกออกจากกันด้วยเครื่องหมาย "\" ดังนั้นถ้าหากผู้อ่านต้องการดึงเฉพาะที่เป็นชื่อไฟล์ของโมดูลเท่านั้น ก็สามารถทำได้โดยใช้เทคนิค ดังตัวอย่างต่อไปนี้

```
Do While InStr(source$, "\")
source$ = Mid$(source$, InStr(source$, "\") + 1)
Loop
```

สำหรับชื่อไฟล์ภายใต้วินโดวส์ 95 นั้นจะแตกต่างกับวินโดวส์ 3.x ดังนั้นก่อนที่จะมีการเรียกใช้ฟังก์ชันนี้ ผู้อ่านก็ควรที่จะตรวจสอบหมายเลขเวอร์ชันของไฟล์ เพื่อดูว่าเป็นไฟล์ที่คอมพายล์สำหรับระบบปฏิบัติการใด ■

GetWindow

การประกาศ Public Declare Function GetWindow Lib "user32" _
(ByVal hwnd As Long, ByVal wCmd As Long) As Long

หน้าที่ รายงานหมายเลข handle ของหน้าต่าง ที่สัมพันธ์กับหน้าต่างต้นทาง

พารามิเตอร์

hwnd	หมายเลข handle ของหน้าต่างต้นทาง	
wCmd	ค่าคงที่ที่ใช้กำหนดความสัมพันธ์กับหน้าต่างต้นทาง ซึ่งสามารถเป็นไปได้ดังต่อไปนี้	
	GW_HWNDFIRST	ค้นหาหน้าต่างที่ระดับสูงสุดของ link-list
	GW_HWNDLAST	ค้นหาหน้าต่างที่ระดับล่างสุดของ link-list
	GW_HWNDNEXT	ค้นหาหน้าต่างถัดไปของ link-list
	GW_HWNDPREV	ค้นหาหน้าต่างที่ผ่านมาของ link-list
	GW_OWNER	ค้นหาเจ้าของหน้าต่างต้นทาง
	GW_CHILD	ค้นหาหน้าต่างลูกลำดับแรกของหน้าต่างต้นทาง

ตัวอย่างการใช้

```
' The desktop is the highest window.
hwnd& = GetDesktopWindow()

' It's first child is the 1st top level window.
hwnd& = GetWindow(hwnd&, GW_CHILD)
```

ค่าที่ส่งกลับ

<> 0	หมายถึง หมายเลข handle ของหน้าต่างที่สัมพันธ์กับหน้าต่างต้นทาง
= 0	หมายถึง เกิดข้อผิดพลาด หรือค้นหาหน้าต่างที่สัมพันธ์กับหน้าต่างต้นทางไม่พบ

หมายเหตุ จากตัวอย่างการใช้งานข้างต้น เป็นการหาหมายเลข handle ของหน้าต่างที่เป็นหน้าต่างลูกของหน้าต่างต้นทาง ซึ่งหน้าต่างต้นทางในที่นี้หมายถึง desktop ของวินโดวส์นั่นเอง

GetWindowLong

การประกาศ Public Declare Function GetWindowLong Lib "user32" _
Alias "GetWindowLongA" (ByVal hwnd As Long, _
ByVal nIndex As Long) As Long

หน้าที่ รายงานข้อมูลเกี่ยวกับโครงสร้างของหน้าต่างที่กำหนด

พารามิเตอร์

hwnd	หมายเลข handle ของหน้าต่าง	
nIndex	ค่าคงที่ที่ใช้กำหนดลักษณะของข้อมูลที่ต้องการ ดังต่อไปนี้	
	GWL_WNDPROC	แอดเดรสของฟังก์ชันวินโดวส์
	GWL_HINSTANCE	หมายเลข handle ของ instance ที่เป็นเจ้าของหน้าต่างที่กำหนด
	GWL_HWNDPARENT	หมายเลข handle ของหน้าต่างแม่ของหน้าต่างที่กำหนด
	GWL_STYLE	รูปแบบของหน้าต่าง (style)
	GWL_EXSTYLE	รูปแบบของหน้าต่าง (extend style)

ตัวอย่างการใช้

```
' Get instance for window
inst& = GetWindowLong(hwnd&, GWL_HINSTANCE)
```

ค่าที่ส่งกลับ

<> 0	หมายถึง ข้อมูลเกี่ยวกับโครงสร้างของหน้าต่างตามที่ถูกกำหนด
= 0	หมายถึง เกิดข้อผิดพลาด

หมายเหตุ ไม่มี

GetWindowText

การประกาศ
```
Public Declare Function GetWindowText Lib "user32" _
Alias "GetWindowTextA" (ByVal hwnd As Long, _
ByVal lpString As String, ByVal cch As Long) As Long
```

หน้าที่ รายงานชื่อ (caption) ของหน้าต่าง หรือข้อความของคอนโทรลพารามิเตอร์

hwnd หมายเลข handle ของหน้าต่าง

lpString สตริงที่มีขนาดเท่ากับ 256 ไบต์ เพื่อใช้เป็นบัฟเฟอร์สำหรับจัดเก็บชื่อ (caption) ของหน้าต่าง

cch กำหนดขนาดของบัฟเฟอร์ lpString - 1 (ในที่นี้เท่ากับ 255 ไบต์)

ตัวอย่างการใช้
```
' Finally, add the window title
     buf$ = String$(256, 0)
     Dummy& = GetWindowText(hwnd&, buf$, 255)
```

ค่าที่ส่งกลับ
> 0 หมายถึง จำนวนไบต์ของชื่อ (caption) ของหน้าต่างที่อ่านได้

= 0 หมายถึง หน้าต่างที่กำหนดไม่มีชื่อ (caption)

หมายเหตุ โดยปกติการใช้คำสั่ง SendKeys ของ VB/Win ในการสลับไปยังแอพพลิเคชันอื่น ๆ นั้น เราจะต้องใช้ caption ของหน้าต่างของแอพพลิเคชันที่ต้องการ กำหนดให้กับคำสั่ง SendKeys ดังนั้นผู้อ่านจึงสามารถอ่านชื่อ (caption) ของหน้าต่างปัจจุบันของแอพพลิเคชันด้วยฟังก์ชันนี้

GetWindowTextLength

การประกาศ
```
Public Declare Function GetWindowTextLength Lib "user32" _
Alias "GetWindowTextLengthA" (ByVal hwnd As Long) _
As Long
```

หน้าที่ พารามิเตอร์ รายงานความยาวของชื่อ (caption) ของหน้าต่าง หรือข้อความของคอนโทรล

hwnd หมายเลข handle ของหน้าต่าง

ตัวอย่างการใช้
```
' Get the window title.
     TitleLen& = GetWindowTextLength(hwnd&)
          buf$ = String$(TitleLen& + 1, 0)
     Dummy& = GetWindowText(hwnd&, buf$, TitleLen&)
```

ค่าที่ส่งกลับ เลขจำนวนเต็ม long หมายถึง ความยาว (จำนวนไบต์) ของชื่อ (caption) ของหน้าต่างที่อ่านได้

หมายเหตุ ไม่มี

GetWindowThreadProcessId

การประกาศ
```
Public Declare Function GetWindowThreadProcessId Lib _
"user32" (ByVal hwnd As Long, lpdwProcessId As Long) As Long
```

หน้าที่ พารามิเตอร์ รายงานหมายเลข ID ของ thread และ process ที่เกี่ยวข้องกับหน้าต่างที่กำหนด

hwnd หมายเลข handle ของหน้าต่าง

lpdwProcessId หมายเลข ID ของ process ที่เป็นเจ้าของหน้าต่าง

ตัวอย่างการใช้
```
' Handling of process in Win32.
     Dummy& = GetWindowThreadProcessId(hwnd, hWndProcess)
```

ค่าที่ส่งกลับ หมายเลข ID ของ thread ที่เป็นเจ้าของหน้าต่างที่กำหนด

หมายเหตุ ฟังก์ชันนี้จะสนับสนุนเฉพาะวินโดวส์ 32 บิต เท่านั้น ทั้งนี้เนื่องจากวินโดวส์ 32 บิต สนับสนุนการทำงานแบบ preemtive-multitasking โดยจะอนุญาตให้แต่ละ thread ของแอพพลิเคชันสามารถทำงานได้พร้อม ๆ กันในเวลาเดียวกัน

SendMessage

การประกาศ	Public Declare Function SendMessage Lib "user32" _ Alias "SendMessageA" (ByVal hwnd As Long, _ ByVal wMsg As Long, ByVal wParam As Long, _ lParam As Long) As Long
หน้าที่	ส่ง message ให้กับวินโดวส์ เพื่อที่จะเรียกใช้ฟังก์ชันวินโดวส์สำหรับหน้าต่างนั้น ๆ
พารามิเตอร์	hwnd หมายเลข handle ของหน้าต่าง wMsg หมายเลขแสดง message wParam พารามิเตอร์ ซึ่งจะมีหรือไม่มีก็ได้ ทั้งนี้ก็ขึ้นกับ message แต่ละชนิด lParam พารามิเตอร์ ซึ่งจะมีหรือไม่มีก็ได้ ทั้งนี้ก็ขึ้นกับ message แต่ละชนิด

ตัวอย่างการใช้	`TabStop&(0) = 30`
	`TabStop&(1) = 94`
	`TabStop&(2) = 236`
	`Dummy& = SendMessage(List1.hwnd, LB_SETTABSTOPS, _`
	`3, TabStop&(0))`

ค่าที่ส่งกลับ	ขึ้นกับชนิดของ message
หมายเหตุ	ฟังก์ชันนี้จะกำหนดให้วินโดวส์จัดการกับหน้าต่างที่กำหนด โดยอาศัยฟังก์ชันวินโดวส์ โดยที่การจัดการแต่ละประเภทนั้นก็ขึ้นกับชนิดของ message เช่น ในตัวอย่างการใช้งาน กำหนดให้ message เท่ากับ LB_SETTABSTOPS ซึ่งจะทำหน้าที่กำหนดตำแหน่งคอลัมน์ (tab) ของแต่ละคอลัมน์ ของคอนโทรล List box เป็นต้น ซึ่งเป็นการทำงานโดยอาศัย message เพื่อสื่อสารกับวินโดวส์

โปรแกรมตัวอย่าง WINCLASS

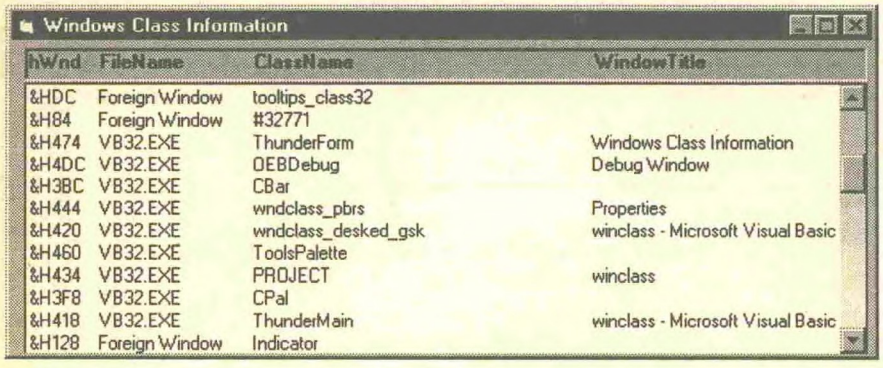
โปรแกรม winclass เป็นโปรแกรมตัวอย่างที่แสดงวิธีการใช้ฟังก์ชันวินโดว์ API ในการอ่านข้อมูลเกี่ยวกับ class ของหน้าต่างทั้งหมดของแอพพลิเคชัน ซึ่งเป็นโปรแกรมง่าย ๆ ที่ประกอบด้วยฟอร์มเพียง 1 ฟอร์ม ดังในรูปที่ 2 โดยจะใช้คอนโทรล Lixt box ในการแสดงผลข้อมูลของแต่ละหน้าต่าง

โดยเมื่อทำการรันโปรแกรมตัวอย่าง ก็จะปรากฏข้อมูลต่าง ๆ ของหน้าต่างที่ถูกจัดการโดยวินโดวส์ดังในรูปที่ 2 ดังเช่น เอนไวรอนเมนต์ของ VB/Win 4.0 นั้นจะมีชื่อคลาสว่า ThumderMain ส่วนหน้าต่าง Debug ของ VB/Win 4.0 นั้นก็จะมีชื่อคลาสว่า OEBDebug เป็นต้น โดยที่คอลัมน์แรกจะแสดงหมายเลข handle ของหน้าต่าง ส่วนคอลัมน์ที่สองจะแสดงชื่อไฟล์ที่เป็นเจ้าของหน้าต่าง คอลัมน์ที่สามจะแสดงชื่อคลาสของแต่ละหน้าต่าง ส่วนคอลัมน์สุดท้ายจะแสดงชื่อ caption ของแต่ละหน้าต่าง

สำหรับการทำงานของโปรแกรม winclass ก็จะเริ่มต้นที่รูทีนเหตุการณ์ Form_Load โดยจะมีการกำหนดตำแหน่งของแต่ละคอลัมน์ของคอนโทรล List1 โดยการใช้ฟังก์ชัน SendMessage ร่วมกับ message ชื่อ LB_SETTABSTOPS ดังนี้

```
Dim TabStop&(3) ' Change to an array of longs
Dim Dummy&

TabStop&(0) = 30
TabStop&(1) = 94
TabStop&(2) = 236
Dummy& = SendMessage(List1.hwnd,
LB_SETTABSTOPS, 3, TabStop&(0))
```



รูปที่ 2
ฟอร์ม Form1 ในขณะออกแบบ

จากนั้น VB/Win ก็จะทำการเรียกใช้รูทีนเหตุการณ์ Form_Activate ซึ่งจะมีการเขียนโค้ดเพื่อเรียกใช้รูทีน LoadTopLevelWindows ซึ่งภายในรูทีนนี้ก็จะเป็นรูทีนหลักของโปรแกรม winclass โดยการทำงานของรูทีนนี้ ก็จะเป็นการเขียนโค้ดเพื่อติดตามแต่ละหน้าต่างของวินโดวส์ โดยการใช้ฟังก์ชัน GetWindow ในการอ่านหมายเลข handle ของหน้าต่างใน link-list ของวินโดว์ ดังนี้

```
' The desktop is the highest window.
hwnd& = GetDesktopWindow()

' It's first child is the 1st top level window.
hwnd& = GetWindow(hwnd&, GW_CHILD)
' Now load all top level windows
Do
        List1.AddItem GetWindowDesc$(hwnd&)
        hwnd& = GetWindow(hwnd&, GW_HWNDNEXT)
Loop While hwnd&
```

เมื่อเราได้หมายเลข handle ของแต่ละหน้าต่างแล้ว ต่อไปก็จะเป็นการอ่านรายละเอียดของแต่ละหน้าต่าง โดยอาศัยฟังก์ชัน GetWindowDesc$ โค้ดภายในฟังก์ชันนี้ก็จะเริ่มต้นด้วยการอ่านหมายเลข ID ของ process ที่เป็นเจ้าของหน้าต่างที่ถูกกำหนดโดยหมายเลข handle (hwnd) โดยการใช้ฟังก์ชัน API ชื่อ GetWindowThreadProcessId จากนั้นก็ทำการเปรียบเทียบหมายเลข ID ที่ได้ กับหมายเลข ID ของกระบวนการ (process) ปัจจุบันที่ได้จากฟังก์ชัน GetCurrentProcessId โดยถ้าหากเป็นหมายเลข ID เดียวกัน ก็แสดงว่าเราสามารถที่จะอ่านชื่อไฟล์ของหน้าต่างนั้น ๆ ได้ นั่นก็หมายความว่าผู้อ่านจะสามารถอ่านชื่อไฟล์ได้เฉพาะของไฟล์ .EXE หรือของหน้าต่างที่ถูกเป็นเจ้าของโดยไฟล์ไลบรารี .DLL หรือ custom control ของ VB/Win เท่านั้น

```
' The windows handle first.
desc$ = "&H" & Hex$(hwnd) & vbTab

' Handling of process in Win32.
Dummy& = GetWindowThreadProcessId
(hwnd, hWndProcess)
If hWndProcess = GetCurrentProcessId() Then
' Get instance for window
inst& = GetWindowLong(hwnd&, GWL_HINSTANCE)

' Get name of source application.
buf$ = String$(256, 0)
' Get the module filename.
Dummy& = GetModuleFileName(inst&, buf$, 255)
buf$ = sZtrim$(GetBaseName(buf$))
Else
        buf$ = "Foreign Window"
End If
' And add it to the description
desc$ = desc$ & buf$ & vbTab
```

ส่วนขั้นตอนการอ่านชื่อไฟล์นั้น ก็ต้องเริ่มต้นด้วยการอ่านหมายเลข ID ของ instance ของหน้าต่างนั้น ๆ เสียก่อนด้วยฟังก์ชัน GetWindowLong จากนั้นจึงอ่านชื่อไฟล์ด้วยฟังก์ชัน GetModuleFile Name อีกครั้ง

```
' next, add the class name
buf$ = String$(256, 0) ' Initialize space again
Dummy& = GetClassName(hwnd&, buf$, 255)
desc$ = desc$ & sZtrim$(buf$) & vbTab

' Finally, add the window title
buf$ = String$(256, 0)
Dummy& = GetWindowText(hwnd&, buf$, 255)
desc$ = desc$ & sZtrim$(buf$)
```

ต่อมาก็จะเป็นโค้ดที่ทำหน้าที่อ่านชื่อคลาสของหน้าต่างโดยใช้ฟังก์ชัน GetClassName และอ่านชื่อ caption ของหน้าต่างโดยใช้ฟังก์ชัน GetWindowText ซึ่งจากการอ่านข้อมูลชนิดสตริงจากฟังก์ชันวินโดว์ API นั้น ผู้อ่านจะเห็นว่าสตริงที่คืนกลับจากฟังก์ชันเหล่านี้จะอยู่ในรูปแบบของภาษาซี หรือเป็นสตริงที่มีการต่อท้ายด้วย ASCII 0 เสมอ ดังนั้นผู้เขียนจึงได้สร้างฟังก์ชัน sZtrim$ เพื่อทำการกำจัด ASCII 0 ออกจากสตริงแล้วจึงนำไปแสดงผล ดังนี้

```
i% = InStr(nullstr$, vbNullChar)
If i% Then
        sZtrim$ = Left$(nullstr$, i% - 1)
Else
        sZtrim$ = nullstr$
End If
```

จากเนื้อหาที่กล่าวมาทั้งหมด ผู้เขียนก็คงจะมีความเข้าใจเกี่ยวกับคลาสของวินโดวส์ และวิธีที่วินโดวส์ใช้ในการติดต่อหรือควบคุมหน้าต่างทั้งหลายได้ดีพอสมควร แต่เนื่องจากประโยชน์ที่จะได้จากการนำคลาสของวินโดวส์ไปใช้นั้นมีมากพอสมควร ผู้อ่านจึงควรที่จะนำโปรแกรม winclass นี้ไปปรับปรุง ก็จะได้รับประโยชน์มากขึ้นครับ สำหรับในตอนหน้าผู้เขียนก็จะไปเจาะลึกถึงฟังก์ชันวินโดว์ API กลุ่มอื่น ๆ ที่สนับสนุนวินโดวส์ 32 บิตกันต่อไป ก็คอยติดตามกันต่อไปนะครับ พบกันใหม่ฉบับหน้าสวัสดีครับ

ลิสติงที่ 1
ไฟล์ WINCLASS.V

```
Form=winclass.Frm
Module=basWinClass; winclass.Bas
ProjWinSize=103,394,248,215
ProjWinShow=2
IconForm="frmWinClass"
Title="winclass"
ExeName32="winclass.exe"
Path32="c:\"
Name="vb4projectWINVIEW"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionComments="Distributed by
Microcomputer Magazine _
Published by SE-ED Co.ltd."
VersionCompanyName="SE-ED Co.,Ltd."
VersionFileDescription="Windows Class Information"
VersionLegalCopyright="Mr.Sutthisak Phongthanapanich"
```

ลิสติงที่ 2
ไฟล์ WINCLASS.B

```
Attribute VB_Name = "basWinClass"
Option Explicit
'
' Project name  : WINCLASS.VBP
' Program Name  : Windows 95 's class
' VB/Win Module : WINCLASS.BAS
' DLL filename  : -
' Copyright     : (c)1997 by Sutthisak Phongthanapanich
'

'_____________________________
'           API contstants
'_____________________________
Public Const GWL_WNDPROC = (-4)
Public Const GWL_HINSTANCE = (-6)
Public Const GWL_HWNDPARENT = (-8)
Public Const GWL_STYLE = (-16)
Public Const GWL_EXSTYLE = (-20)

Public Const GW_HWNDFIRST = 0
Public Const GW_HWNDLAST = 1
Public Const GW_HWNDNEXT = 2
Public Const GW_HWNDPREV = 3
Public Const GW_OWNER = 4
Public Const GW_CHILD = 5

' windows control message.
Public Const LB_SETTABSTOPS = &H192

Public Declare Function GetClassName Lib "user32" Alias "GetClassNameA" _
               (ByVal hwnd As Long, ByVal lpClassName As String, _
               ByVal nMaxCount As Long) As Long
Public Declare Function GetCurrentProcessId Lib "kernel32" () As Long
Public Declare Function GetDesktopWindow Lib "user32" () As Long
Public Declare Function GetModuleFileName Lib "kernel32" Alias _
               "GetModuleFileNameA" (ByVal hModule As Long, _
               ByVal lpFileName As String, ByVal nSize As Long) As Long
Public Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, _
               ByVal wCmd As Long) As Long
Public Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" _
               (ByVal hwnd As Long, ByVal nIndex As Long) As Long
Public Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" _
               (ByVal hwnd As Long, ByVal lpString As String, _
               ByVal cch As Long) As Long
Public Declare Function GetWindowThreadProcessId Lib "user32" _
               (ByVal hwnd As Long, lpdwProcessId As Long) As Long
Public Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
                  (ByVal hwnd As Long, ByVal wMsg As Long,ByVal wParam As Long, _
                  lParam As Long) As Long
```

ลิสติงที่ 3
ไฟล์ WINCLASS.FRM

```
VERSION 4.00
Begin VB.Form frmWinClass
   Caption         =   "Windows Class Information"
   ClientHeight    =   2925
   ClientLeft      =   690
   ClientTop       =   2070
   ClientWidth     =   8100
   BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
   Name            =   "MS Sans Serif"
   Size            =   8.25
   Charset         =   222
   Weight          =   700
   Underline       =   0   'False
   Italic          =   0   'False
   Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H80000008&
   Height          =   3330
   Left            =   630
   LinkMode        =   1  'Source
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   ScaleHeight     =   195
   ScaleMode       =   3  'Pixel
   ScaleWidth      =   540
   Top             =   1725
   Width           =   8220
   Begin VB.ListBox List1
      Appearance      =   0  'Flat
      BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
         Name            =   "MS Sans Serif"
         Size            =   8.25
         Charset         =   222
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   2550
      ItemData        =   "winclass.frx":0000
      Left            =   120
      List            =   "winclass.frx":0002
      TabIndex        =   1
      Top             =   390
      Width           =   7935
   End
   Begin VB.Label Label1
      BorderStyle     =   1  'Fixed Single
      Caption         =   "hWnd   FileName          ClassName
WindowTitle"
      ForeColor       =   &H00FF0000&
      Height          =   315
      Left            =   120
      TabIndex        =   0
      Top             =   60
      Width           =   7935
   End
End
Attribute VB_Name = "frmWinClass"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
'
' Project name  : WINCLASS.VBP
' Program Name  : Windows 95 's class
' VB/Win Module : WINCLASS.FRM
' DLL filename  : -
' Copyright     : (c)1997 by Sutthisak Phongthanapanich
'

Private Sub Form_Activate()
    Refresh
    LoadTopLevelWindows
End Sub
```

ลิสติงที่ 3 (ต่อ)

```
Private Sub Form_Load()
    Dim TabStop&(3) ' Change to an array of long
    Dim Dummy&

    TabStop&(0) = 30
    TabStop&(1) = 94
    TabStop&(2) = 236
    Dummy& = SendMessage(List1.hwnd, LB_SETTABSTOPS, 3, TabStop&(0))
End Sub

Private Function GetBaseName$(ByVal source$)
    Dim i&

    Do While InStr(source$, "\")
        source$ = Mid$(source$, InStr(source$, "\") + 1)
    Loop

    i& = InStr(source$, ":")
    If i& Then source$ = Mid$(source$, i& + 1)

    GetBaseName$ = source$
End Function

Private Function GetWindowDesc$(hwnd&)
    Dim desc$, buf$
    Dim inst&, hWndProcess&
    Dim Dummy&

    ' The windows handle first.
    desc$ = "&H" & Hex$(hwnd) & vbTab

    ' Handling of process in Win32.
    Dummy& = GetWindowThreadProcessId(hwnd, hWndProcess)
    If hWndProcess = GetCurrentProcessId() Then
        ' Get instance for window
        inst& = GetWindowLong(hwnd&, GWL_HINSTANCE)

        ' Get name of source application.
        buf$ = String$(256, 0)
        ' Get the module filename.
        Dummy& = GetModuleFileName(inst&, buf$, 255)
        buf$ = sZtrim$(GetBaseName(buf$))
    Else
        buf$ = "Foreign Window"
    End If
    ' And add it to the description
    desc$ = desc$ & buf$ & vbTab

    ' next, add the class name
    buf$ = String$(256, 0) ' Initialize space again
    Dummy& = GetClassName(hwnd&, buf$, 255)
    desc$ = desc$ & sZtrim$(buf$) & vbTab


    ' Finally, add the window title
    buf$ = String$(256, 0)
    Dummy& = GetWindowText(hwnd&, buf$, 255)
    desc$ = desc$ & sZtrim$(buf$)

    ' And return the description
    GetWindowDesc$ = desc$
End Function


'   Loads the listbox with a list of all top level
'   windows.
'
Private Sub LoadTopLevelWindows()
    Dim hwnd&

    List1.Clear

    ' The desktop is the highest window.
    hwnd& = GetDesktopWindow()
```

→

ลิสติงที่ 3 (ต่อ)

```
    ' It's first child is the 1st top level window.
    hwnd& = GetWindow(hwnd&, GW_CHILD)
    ' Now load all top level windows
    Do
        List1.AddItem GetWindowDesc$(hwnd&)
        hwnd& = GetWindow(hwnd&, GW_HWNDNEXT)
    Loop While hwnd&
End Sub

Private Function sZtrim$(nullstr$)
    Dim i%

    i% = InStr(nullstr$, vbNullChar)
    If i% Then
        sZtrim$ = Left$(nullstr$, i% - 1)
    Else
        sZtrim$ = nullstr$
    End If
End Function
```

เทคนิคประจำคอลัมน์

ถาม OLE คอนโทรล แตกต่างกับคอนโทรลของ VB/Win (*.OCX) อย่างไร ?

ตอบ คอนโทรลของ VB/Win เช่น Picture box หรือ Text box เป็นต้น จัดว่าเป็น OLE คอนโทรลประเภทหนึ่ง ที่มีหน้าต่างเป็นของตัวเอง ดังนั้นคอนโทรลของ VB/Win จึงมีคุณสมบัติ hWnd เพื่อรายงานหมายเลข handle ของหน้าต่างของคอนโทรล ดังนั้นผู้อ่านจึงสามารถใช้ฟังก์ชันวินโดว์ API ที่เกี่ยวข้องกับคลาส ตามที่ปรากฏในบทความตอนนี้กับคอนโทรลของ VB/Win ได้อย่างไม่มีปัญหา แต่ยังมี OLE คอนโทรลบางประเภทที่มีหน้าต่างเฉพาะในขณะที่แสดงผลและอยู่ในสถานะ active เท่านั้น ดังนั้นในขณะที่ไม่มีการแสดงผลของ OLE คอนโทรล หน้าต่างก็จะถูกทำลาย ดังนั้นหมายเลข handle ของหน้าต่างของ OLE คอนโทรลประเภทนี้ ก็จะไม่เป็นจริง ซึ่งโดยปกติเราจะตรวจสอบหมายเลข handle ของ OLE คอนโทรล ว่าจะแสดงถึงหน้าต่างที่เป็นจริงหรือไม่ ก็จะทำการตรวจสอบด้วยฟังก์ชันวินโดว์ API ที่ชื่อ IsWindow สำหรับการอ่านหมายเลข handle ของ OLE คอนโทรล ผู้อ่านก็สามารถทดลองได้จากโปรแกรมตัวอย่าง winclass.vbp

ตอบปัญหาชิงรางวัล

มีคำถามอยู่ 10 คำถามที่อยากให้ท่านผู้อ่านช่วยส่งความหมายพร้อมยกตัวอย่างประกอบสั้น ๆ ประกอบการอธิบาย ผู้ที่ตอบถูกจะได้รับของขวัญเป็นหนังสือตำราด้านคอมพิวเตอร์

1. GigaPop
2. RSVP
3. RTP
4. ATM
5. QoS
6. SONET
7. Internet 2
8. Terrabyte
9. FPGA
10. IPv6

ส่งคำตอบมาที่ **กองบรรณาธิการ นิตยสารไมโครคอมพิวเตอร์** บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) 800/43-45 ซอยตระกูลสุข ถนนอโศก-ดินแดง ดินแดง กรุงเทพฯ 10320 (วงเล็บมุมซองตอบปัญหาชิงรางวัล)